**Assignment of EAD**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**Submitted to:**

Dr.Shuja-ur-Rehman Baig

**Submitted by:**

Saad Ishtiaq

Roll no: BSEF18A025

***(DO NOT COPY.*** *This property belongs to***Saad Ishtiaq.** Beware***!!! )***

**Punjab University College of Information Technology,**

**PUCIT (Old Campus),**

**LAHORE**

**EAD HOMEWORK\_1**

Information is taken from

Provided Reference Book , Geeks For Geeks and Google.

**(BSSE & BSIT FALL 2018)**

**Q1: What is IL and how IL code is converted into machine executable code?**

**Ans:** IL stands for **intermediate language code**. Source code is converted by compiler into intermediate language code. IL is independent of operating system and underlying hardware.

**Roslyn the C# compiler** used by the **dotnet** CLI tool converts C# source code into intermediate language (IL) code and stores the IL in an assembly (a DLL or EXE file). IL code statements are like assembly language instructions, which are executed by .NET Core's virtual machine, known as CoreCLR. At runtime, CoreCLR loads the IL code from the assembly, the just-in-time (JIT) compiler compiles it into native CPU instructions, and then it is executed by the CPU of machine.

**Q2: Differentiate between .NET Core and .NET Frameworks. (Minimum 4 Points)**

**Ans:**

|  |  |  |
| --- | --- | --- |
| Based on | **.NET Core** | **.NET Frameworks** |
| **Open Source** | .Net Core is an open source. | Certain components of the .Net Framework are open source. |
| **Performance and Scalability** | .NET Core offers high performance and scalability. | .Net Framework is less effective in comparison to .Net Core in terms of performance and scalability of applications. |
| **Compatibility** | .NET Core is compatible with various operating systems — Windows, Linux, and Mac OS. | .NET Framework is compatible only with the Windows operating system. |
| **CLI Tools** | .NET Core provides light-weight editors and command-line tools for all supported platforms. | .Net Framework is heavy for Command Line Interface and developers prefer to work on the lightweight Command Line Interface. |

**Q3: Why Decimal data type variable takes more space and store less data than double type variable? Please explain it with example.**

**Ans**: The main difference is Double is **binary floating point types** and a Decimal will store the value as a **floating decimal point type**. So Decimals have much higher precision and Double, cannot accurately represent all numbers. But in performance wise Decimals are slower than double.

**Double -** 64 bit **(15-16** digits**) Binary Floating(**10001.10011001011**)**

**Decimal -** 128 bit **(28-29** significant digits**) Floating decimal (**12345.65789**)**

**Example From Book:**

Console.WriteLine("Using doubles:");

double a = 0.1;

double b = 0.2;

if (a + b == 0.3)

{

Console.WriteLine($"{a} + {b} equals 0.3");

}

else

{

Console.WriteLine($"{a} + {b} does NOT equal 0.3");

}

/\*------output: 0.1 + 0.2 does NOT equal 0.3-----------\*/

Console.WriteLine("Using decimals:");

decimal z = 0.1M; // M suffix means a decimal literal value

decimal y = 0.2M;

if (z + y == 0.3M)

{

Console.WriteLine($"{z} + {y} equals 0.3");

}

else

{

Console.WriteLine($"{z} + {y} does NOT equal 0.3");

}

/\*--------------output: 0.1 + 0.2 equals 0.3-------------------------------------\*/

In above example we needed precision so decimal gave us the desired answer, If we need accuracy then we will use the double varaible type.

**Q4: What are the default values of**

**1. Int** : 0

**2. uint** : 0

**3. Double** : 0

**4. Float** : 0

**5. Decimal** : 0

**6. Object** : null

**Q5: Please find out the errors if they have in these snippets and rewrite the code.**

**1: static void Main(string[] args)**

**{**

**string String = null;**

**int? numbers = 85;**

**String = $"Ahmad got [numbers] marks/n";**

**object obj = String;**

**obj = obj.ToLower();**

**Console.writeLine(obj);**

**}**

**Sol:**

static void Main(string[] args)

{

string? str = null;

int? numbers = 85;

str = $"Ahmad got {numbers} marks\n";

object obj = str;

obj = obj. ToString().ToLower();

Console.WriteLine(obj);

}

**2: static void Main(string[] args)**

**{**

**int age = Console.Read();**

**Console.writeLine(age);**

**}**

**Sol:**

{

int age = Convert.ToInt32( Console.ReadLine() );

Console.WriteLine(age);

}

**Q6: Differentiate between var and dynamic with an example.**

**Ans:**

**var type** of the variable is automatically deduced at compile time by the compiler from the value used to initialize the variable. Var does not allow the type of variable to be changed once assigned. Var cannot be used for property or return values from a function.

Examples:

var a = 83;

var b = "Saad";

Console.WriteLine(a.GetType());

Console.WriteLine(b.GetType());

/\*--------------------output:

System.Int32

System.String

------------------------\*/

Whereas, a **dynamic type** is used to avoid the compile-time type checking. The compiler does not check the type of the dynamic type variable at compile time, instead of this, the compiler gets the type at the run time.It allows the type of variable to be changed once it is assigned. It can also be used to create properties and return values from a function.

Examples:

dynamic a = 83;

dynamic b = "Saad";

Console.WriteLine(a.GetType());

Console.WriteLine(b.ToLower());

/\*---------------output:

System.Int32

saad

-------------------------\*/

**Q7:Write a program that displays the ascii value of a key when pressed by user.**

**Code:**

using System;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Press any Key: ");

ConsoleKeyInfo key = Console.ReadKey();

Console.WriteLine();

Console.WriteLine(

format: "Character: {0} \nModifier: {1} \nAscii value of a key: {2}",

arg0: key.KeyChar,

arg1: key.Modifiers,

arg2: (int)key.KeyChar

);

}

}

}

**Q8: Write a program that displays the table of a number X to the length N. Your program must get these values as arguments in the main function. You need to use formatted output in your program to get full marks.**

**Code:**

**Sol No 1: Input as an argument to main function**

**Run :** dotnet run 2 10

using System;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

int number = Convert.ToInt32(args[0]);

int length = Convert.ToInt32(args[1]);

Console.WriteLine("Number: " + number);

Console.WriteLine("Lemgth of table: " + length);

for (int i = 1; i <= length; i++)

{

Console.WriteLine(

format: "{0} x {1} = {2}",

arg0: number,

arg1: i,

arg2: number \* i

);

}

}

}

}

**Sol No 2: Input from console**

using System;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter a Number: ");

int number = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter lemgth of table: ");

int length = Convert.ToInt32(Console.ReadLine());

for (int i = 1; i <= length; i++)

{

Console.WriteLine(

format: "{0} x {1} = {2}",

arg0: number,

arg1: i,

arg2: number \* i

);

}

}

}

}

**Q9: Write a program that takes 5 Fruits names along with their prices in command line arguments and display them in proper output formatting.**

**Code:**

**Sol No 1: Input as an argument to main function**

**Run :** dotnet run mango 34 apple 45 banana 67 peach 99 grapes 100

using System;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

string[] fruits = new string[5];

double[] price = new double[5];

int j = 0;

for (int i = 0; i < 10; i++)

{

fruits[j] =args[i];

price[j] = Convert.ToDouble( args[i=i+1]);

j++;

}

for (int i = 0; i < 5; i++)

{

Console.WriteLine(

format: "{0,-9} costs {1,9:C}",

arg0: fruits[i],

arg1: price[i]

);

}

}

}

}

**Sol No 2: Input from console**

using System;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

string[] fruits = new string[5];

double[] price = new double[5];

for (int i = 0; i < 5; i++)

{

Console.WriteLine("Enter Fruit Name: ");

fruits[i] = Console.ReadLine();

Console.WriteLine("Enter Fruit Price: ");

price[i] = Convert.ToDouble(Console.ReadLine());

}

for (int i = 0; i < 5; i++)

{

Console.WriteLine(

format: "{0,-9} costs {1,9:C}",

arg0: fruits[i],

arg1: price[i]

);

}

}

}

}